PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-149008
(43)Date of publication of application : 30.05.2000

(51)Int.CI. G06T 3/40
H04N 1/387
H04N 1/409

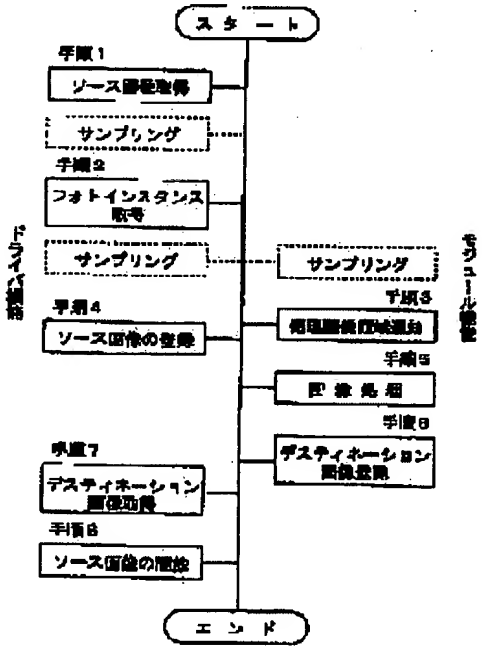
(21)Application number : 11-243399
(22)Date of filing : 30.08.1999

(71)Applicant : SEIKO EPSON CORP
(72)Inventor : ISHIDA MASANORI

(30)Priority
Priority number : 10245337 Priority date : 31.08.1998 Priority country : JP

(54) MEDIUM RECORDING IMAGE PROCESSING PROGRAM AND IMAGE PROCESSING DEVICE AND METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a medium which records an image processing program that can prevent cases, where the image processing cannot be executed according to a higher level of image processing technique and also to provide the image processing device and method.
SOLUTION: A module notifies the driver of a processing image area (procedure 3) after a source image is acquired (procedure 1) and after a photo instance acquisition processing (procedure 2). At the same time, the necessary fringing areas are added with every image processing module, so that the source image (S-unit) is registered by the driver after the necessary fringing area is added to a requested processing image area (procedure 4). Based on a registered S-unit, image processings are successively executed by utilizing the enhancement and enlargement processings (procedure 5). Thus, it is possible to secure and process extremely simply a necessary area in an, even when the image processing requires a fringing area.



LEGAL STATUS
[Date of request for examination] 11.04.2000

[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3467727
[Date of registration]	05.09.2003
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－149008
（P2000－149008A）
(43)公開日　平成12年５月30日(2000.5.30)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'（参考）
Ｇ０６Ｔ　3/40		Ｇ０６Ｆ　15/66	３５５Ｃ	
Ｈ０４Ｎ　1/387		Ｈ０４Ｎ　1/387		
1/409		1/40	１０１Ｄ	

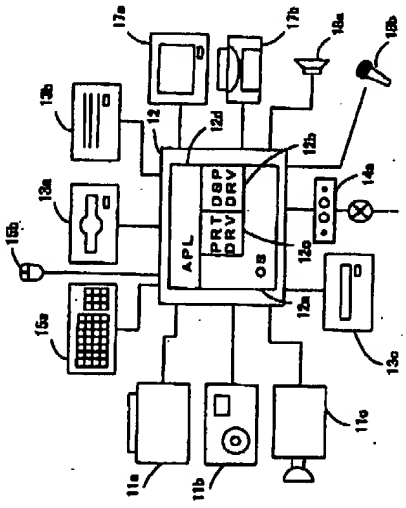
審査請求　未請求　請求項の数７　ＯＬ　（全 20 頁）

(21)出願番号　特願平11－243399
(22)出願日　平成11年８月30日(1999.8.30)
(31)優先権主張番号　特願平10－245337
(32)優先日　平成10年８月31日(1998.8.31)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000002369
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号
(72)発明者　石田　正紀
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内
(74)代理人　100096703
弁理士　横井　俊之

(54)【発明の名称】　画像処理プログラムを記録した媒体、画像処理装置および画像処理方法

(57)【要約】
【課題】　既存の画素についての処理を実行するためにはその周縁の画素についての情報を取り入れておかなければならない場合もあり、そのような場合だと画像領域の周縁部分で必要な画素の情報を参照できなくなるという課題があった。
【解決手段】　ソース画像の取得処理後（手順１）、フォトインスタンス取得処理（手順２）を経たら、モジュールの側からドライバの側に向けて処理画像領域を通知するが（手順３）、このときに各画像処理モジュールで必要な周縁領域を加えていくことにより、要求された処理画像領域に必要な周縁領域を加えた上でソース画像（Ｓ－Ｕｎｉｔ）をドライバの側が登録し（手順４）、これを利用して順次画像処理（強調処理と拡大処理（手順５））を実行することになり、画像処理が周縁領域を必要とする場合にも極めてシンプルに必要領域を確保して処理を実行することができる。



1

【特許請求の範囲】

【請求項１】　ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理をコンピュータに実行させる画像データ補間処理プログラムを記録した媒体であって、

同画像処理プログラムは、上記オブジェクト画像の分割オブジェクトデータが入力されるドライバと、各オブジェクト画像ごとに画像処理を実行するモジュールとを具備し、かつ、上記ドライバは上記分割オブジェクトデータをスプールするとともに、上記モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、モジュールが処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項２】　上記請求項１に記載の画像処理プログラムを記録した媒体において、上記モジュールが個別の画像処理を実行する複数の画像処理オブジェクトを有する場合に、各画像処理オブジェクトの画像処理において必要とする周縁領域を予め加えておいた画像領域を上記ドライバがメモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項３】　上記請求項２に記載の画像処理プログラムを記録した媒体において、上記モジュールの各画像処理オブジェクトは順次自己の画像処理において必要とする周縁領域を処理対象とする画像領域に加えて残りの画像処理オブジェクトに通知していき、最後の画像処理オブジェクトにおいて周縁領域を加えられた画像領域を上記ドライバがメモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項４】　上記請求項１～請求項３のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、周縁領域を加えた画像領域を含めた上で分割オブジェクトデータ単位でメモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項５】　上記請求項１～請求項４のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、メモリ上に展開した分割オブジェクトデータのうちで利用頻度の高いものを限定して利用可能にすることを特徴とする画像処理プログラムを記録した媒体。

【請求項６】　ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理装置であって、

上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理手段と、

各オブジェクト画像ごとに画像処理を実行するデータ処理手段とを具備し、

かつ、上記データ管理手段は上記分割オブジェクトデータをスプールするとともに、上記データ処理手段が処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理手段が処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開することを特徴とする画像処理装置。

【請求項７】　ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理方法であって、

上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理工程と、

各オブジェクト画像ごとに画像処理を実行するデータ処理工程とを具備し、

上記データ管理工程では、上記分割オブジェクトデータをスプールするとともに、上記データ処理工程で処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理工程で処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開することを特徴とする画像処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、画像処理プログラムを記録した媒体、画像処理装置および画像処理方法に関する。

【０００２】

【従来の技術】コンピュータなどでは、画像をドットマトリクス状に配置された画素で表しているが、近年、仮想的なドットマトリクス状の領域に対して、オブジェクト画像を適宜配置して一枚の画像を表す手法が利用されている。すなわち、文字だけを表すオブジェクト画像や、イメージを表すオブジェクト画像を個別に用意し、所定の順番で重ね合わせて画像を完成させる。このような画像を扱う際、コンピュータの処理の都合上、一つのオブジェクト画像を複数に分割することが多い。例えば、オブジェクト画像が大きくなるのにつれてデータ量も極めて多大となり、一体として処理するには処理資源の負担が大きくなるからである。

【０００３】一方、コンピュータで複数の段階に分けて画像を処理することも可能であり、アプリケーションのように内部において自由に処理できるものの他、オペレーティングシステムの一部として機能するドライバやモジュールによってメモリなどの制約を強く受けながら処理しなければならないものもある。すなわち、分割され

たオブジェクトデータを連結して元のオブジェクト画像を生成することなく、画像処理をしなければならないものがある。従来、モジュールがオブジェクトデータの処理を実行する場合にドライバに対してオブジェクトデータへのアクセスを要求するが、それはモジュールで処理しようとしている画像領域に過ぎなかった。

【０００４】

【発明が解決しようとする課題】上述した従来のものにおいては、オブジェクトデータがモジュールに提供するのは本来の処理対象である画像領域に対応したものであった。しかし、画像処理が既存の単独画素だけに基づいて処理を実行しているときには問題が生じないものの、既存の画素についての処理を実行するためにはその周縁の画素についての情報を取り入れておかなければならない場合もあり、そのような場合だと画像領域の周縁部分で必要な画素の情報を参照できなくなるという課題があった。

【０００５】本発明は、上記課題にかんがみてなされたもので、画像処理の高度化に伴って画像処理を実行できなくするのを防止することが可能な画像処理プログラムを記録した媒体、画像処理装置および画像処理方法の提供を目的とする。

【０００６】

【課題を解決するための手段】上記目的を達成するため、請求項１にかかる発明は、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理を実行する画像処理プログラムを記録した媒体であって、同画像処理プログラムは、上記オブジェクト画像の分割オブジェクトデータが入力されるドライバと、各オブジェクト画像ごとに画像処理を実行するモジュールとを具備し、かつ、上記ドライバは上記分割オブジェクトデータをスプールするとともに、上記モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、モジュールが処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開する構成としてある。

【０００７】上記のように構成した請求項１にかかる発明においては、ドライバにオブジェクト画像の分割オブジェクトデータが入力されると、モジュールが各オブジェクト画像ごとに画像処理を実行するが、ここで上記ドライバは上記分割オブジェクトデータをスプールするとともに、上記モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能としている。そして、この際、同ドライバは同モジュールが処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開している。すなわち、本画像処理プログラムでは、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理を実行するが、処理の都合上、ドライバは分割オブジェクトデータをスプールし、モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開しており、このときにメモリ上に展開される領域は本来の処理対象とする画像領域だけでなく、モジュールで実行する画像処理において参照する周縁領域まで加えてある。

【０００８】ところで、モジュールは必ずしも単独の画像処理を実行するものに限られる必要はなく、複数の画像処理を実行できるものであっても構わない。このような場合に好適な一例として、請求項２にかかる発明は、請求項１に記載の画像処理プログラムを記録した媒体において、上記モジュールが個別の画像処理を実行する複数の画像処理オブジェクトを有する場合に、各画像処理オブジェクトの画像処理において必要とする周縁領域を予め加えておいた画像領域を上記ドライバがメモリ上に展開する構成としてある。上記のように構成した請求項２にかかる発明においては、複数の画像処理オブジェクトにおけるそれぞれの画像処理において必要とする周縁領域を予め加えておき、上記ドライバはかかる周縁領域を含めた画像領域をメモリ上に展開している。

【０００９】このように各画像処理オブジェクトの画像処理において必要とする周縁領域を予め加えておくにあたり、請求項３にかかる発明は、請求項２に記載の画像処理プログラムを記録した媒体において、上記モジュールの各画像処理オブジェクトは順次自己の画像処理において必要とする周縁領域を処理対象とする画像領域に加えて残りの画像処理オブジェクトに通知していき、最後の画像処理オブジェクトにおいて周縁領域を加えられた画像領域を上記ドライバがメモリ上に展開する構成としてある。上記のように構成した請求項３にかかる発明においては、各画像処理オブジェクトが自己の画像処理において必要とする周縁領域を処理対象とする画像領域に加えることができ、複数の画像処理オブジェクトがある場合に、これを数珠つなぎとして、それぞれの画像処理オブジェクトが次段の画像処理オブジェクトに対して周縁領域を加えた画像領域を通知していく。これにより、最後の画像処理オブジェクトにおいて周縁領域を加えられた画像領域を参照すれば、全ての画像処理オブジェクトにおいて必要となる周縁領域が加えられた画像領域が分かり、ドライバはかかる最終の画像領域をメモリに展開すればよい。

【００１０】さらに、請求項４にかかる発明は、請求項１～請求項３のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、周縁領域を加

えた画像領域を含めた上で分割オブジェクトデータ単位でメモリ上に展開する構成としてある。メモリ上に展開されるオブジェクトデータは必ずしも処理対象となる画像領域そのままである必要はなく、余裕を持って展開されても構わない。すなわち、参照される周縁領域も含めてメモリに展開されていなければ困るが、逆に余分に展開されていることは支障がない。このため、上記のように構成した請求項４にかかる発明においては、周縁領域を加えた画像領域を含めた上でドライバが分割オブジェクトデータ単位でメモリ上に展開する。

【００１１】分割オブジェクトデータの数が極めて多くなると、処理の低下を免れなくなるが、そのような場合に好適な一例として、請求項５にかかる発明は、請求項１～請求項４のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、メモリ上に展開した分割オブジェクトデータのうちで利用頻度の高いものを限定して利用可能にする構成としてある。メモリ上に展開した分割オブジェクトデータは必ずしも全てを必要としないこともあるため、上記のように構成した請求項５にかかる発明においては、利用頻度の高い分割オブジェクトデータを限定して利用可能にすることによって不要な分割オブジェクトデータを処理対象とすることなく、処理速度などの低下を防止する。

【００１２】以上においては、発明の思想の具現化例として、ソフトウェアを記録した記録媒体について説明した。むろん、その記録媒体は、磁気記録媒体であってもよいし光磁気記録媒体であってもよいし、今後開発されるいかなる記録媒体においても全く同様に考えることができる。また、一次複製品、二次複製品などの複製段階については全く問う余地無く同等である。その他、供給方法として通信回線を利用して行なう場合でも本発明が利用されていることにはかわりない。さらに、一部がソフトウェアであって、一部がハードウェアで実現されている場合においても発明の思想において全く異なるものではなく、一部を記録媒体上に記憶しておいて必要に応じて適宜読み込まれるような形態のものとしてあってもよい。

【００１３】しかしながら、本発明がかかるソフトウェアとしてのみ利用されるというわけでないことは明らかであり、その一例として、請求項６にかかる発明は、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理装置であって、上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理手段と、各オブジェクト画像ごとに画像処理を実行するデータ処理手段とを具備し、かつ、上記データ管理手段は上記分割オブジェクトデータをスプールするとともに、上記データ処理手段が処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理手段が処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開する構成としてある。上記のように構成した請求項６にかかる発明における動作は、ソフトウェアの場合と同様である。

【００１４】また、このような手法は必ずしも実体のある装置に限られる必要はなく、その方法としても機能することは容易に理解できる。このため、請求項７にかかる発明は、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理方法であって、上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理工程と、各オブジェクト画像ごとに画像処理を実行するデータ処理工程とを具備し、上記データ管理工程では、上記分割オブジェクトデータをスプールするとともに、上記データ処理工程で処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理工程で処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開する構成としてある。

【００１５】

【発明の効果】以上説明したように本発明は、画像処理の高度化に伴って処理対象の画像領域の周縁領域まで参照する必要が生じる場合にも画像処理を実行することが可能な画像処理プログラムを記録した媒体を提供することができる。また、請求項２にかかる発明によれば、複数の画像処理によって参照する周縁領域が異なる場合も画像処理を実行することができる。さらに、請求項３にかかる発明によれば、各画像処理オブジェクトが順次周縁領域を通知していくことにより、必要な周縁領域の管理を行う煩わしさを減少させることができる。さらに、請求項４にかかる発明によれば、　ドライバはオブジェクトデータ単位でスプールしているため、さらに必要領域を考慮して切り出すということが必要でなくなり、処理速度は向上する。さらに、請求項５にかかる発明によれば、利用頻度の低い分割オブジェクトデータを処理対象としないことによって処理速度の低下を防止することができる。さらに、請求項６にかかる発明によれば、同様の効果を奏することが可能な画像処理装置を提供でき、請求項７にかかる発明によれば、画像処理方法を提供できる。

【００１６】

【発明の実施の形態】以下、図面にもとづいて本発明の実施形態を説明する。図１は本発明の一実施形態にかかる画像処理プログラムを適用した画像処理システムを実現するハードウェアの一例としてのコンピュータシステ

ム１０をブロック図により示している。本コンピュータシステム１０は、画像データを直接的に入力する画像入力デバイスとして、スキャナ１１ａとデジタルスチルカメラ１１ｂとビデオカメラ１１ｃとを備えており、コンピュータ本体１２に接続されている。それぞれの入力デバイスは画像をドットマトリクス状の画素で表現した画像データを生成してコンピュータ本体１２に出力可能となっており、ここで同画像データはＲＧＢの三原色においてそれぞれ２５６階調表示することにより、約１６７０万色を表現可能となっている。

【００１７】コンピュータ本体１２には、外部補助記憶装置としてのフロッピーディスクドライブ１３ａとハードディスク１３ｂとＣＤ－ＲＯＭドライブ１３ｃとが接続されており、ハードディスク１３ｂにはシステム関連の主要プログラムが記録されており、フロッピーディスクやＣＤ－ＲＯＭなどから適宜必要なプログラムなどを読み込み可能となっている。また、コンピュータ本体１２を外部のネットワークなどに接続するための通信デバイスとしてモデム１４ａが接続されており、外部のネットワークに同公衆通信回線を介して接続し、ソフトウェアやデータをダウンロードして導入可能となっている。この例ではモデム１４ａにて電話回線を介して外部にアクセスするようにしているが、ＬＡＮアダプタを介してネットワークに対してアクセスする構成とすることも可能である。

【００１８】ここで、外部補助記憶装置のうち、フロッピーディスクドライブ１３ａやＣＤ－ＲＯＭドライブ１３ｃについては、記録媒体自身が交換可能であり、この記録媒体に画像データが記録された状態で供給されることにより、画像入力デバイスの一手段ともなりうる。また、モデム１４ａやＬＡＮアダプタを介してネットワークにアクセスした場合、このネットワークから画像データが供給されることもあり、このような場合も画像入力デバイスの一手段となりうる。この他、コンピュータ本体１２の操作用にキーボード１５ａやポインティングデバイスとしてのマウス１５ｂも接続され、さらに、マルチメディア対応のためにスピーカ１８ａやマイク１８ｂを備えている。一方、画像出力デバイスとして、ディスプレイ１７ａとカラープリンタ１７ｂとを備えている。ディスプレイ１７ａについては水平方向に８００画素と垂直方向に６００画素の表示エリアを備えており、各画素毎に上述した１６７０万色の表示が可能となっている。むろん、この解像度は一例に過ぎず、６４０×４８０画素であったり、１０２４×７６８画素であるなど、適宜、変更可能である。

【００１９】また、印刷装置としてのカラープリンタ１７ｂはインクジェットプリンタであり、ＣＭＹＫの四色の色インクを用いて記録媒体たる印刷用紙上にドットを付して画像を印刷可能となっている。画像密度は３６０×３６０ｄｐｉや７２０×７２０ｄｐｉといった高密度印刷が可能となっているが、階調表限については色インクを付すか否かといった２階調表現となっている。色インクについては、かかる四色のものに限らず、色の薄いライトシアンやライトマゼンタを加えた六色によってドットの目立ちを低減させることも可能であるしインクジェット方式に限らずカラートナーを利用した静電写真方式などを採用することも可能である。このような画像入力デバイスを使用して画像を入力しつつ、画像出力デバイスに表示あるいは出力するため、コンピュータ本体１２内では所定のプログラムが実行されることになる。そのうち、基本プログラムとして稼働しているのはオペレーティングシステム（ＯＳ）１２ａであり、このオペレーティングシステム１２ａにはディスプレイ１７ａでの表示を行わせるディスプレイドライバ（ＤＳＰＤＲＶ）１２ｂとカラープリンタ１７ｂに印刷出力を行わせるプリンタドライバ（ＰＲＴ　ＤＲＶ）１２ｃが組み込まれている。これらのドライバ１２ｂ，１２ｃの類はディスプレイ１７ａやカラープリンタ１７ｂの機種に依存しており、それぞれの機種に応じてオペレーティングシステム１２ａに対して追加変更可能である。また、機種に依存して標準処理以上の付加機能を実現することもできるようになっている。すなわち、オペレーティングシステム１２ａという標準システム上で共通化した処理体系を維持しつつ、許容される範囲内での各種の追加的処理を実現できる。

【００２０】この基本プログラムとしてのオペレーティングシステム１２ａ上でアプリケーション１２ｄが実行される。アプリケーション１２ｄの処理内容は様々であり、操作デバイスとしてのキーボード１５ａやマウス１５ｂの操作を監視し、操作された場合には各種の外部機器を適切に制御して対応する演算処理などを実行し、さらには、処理結果をディスプレイ１７ａに表示したり、カラープリンタ１７ｂに出力したりすることになる。かかるコンピュータシステム１０では、画像入力デバイスであるスキャナ１１ａなどで写真などを読み取って画像データを取得することができる他、デジタルスチルカメラ１１ｂで撮影した画像データを取得したり、ビデオカメラ１１ｃで撮影した動画としての画像データを取得することができる。このような画像データは最終的に画像出力デバイスとしてのディスプレイ１７ａで表示したり、カラープリンタ１７ｂで印刷することになるが、元の画像データのままでは写りが悪いなどの問題があることが多く、そのような場合には、何らかの修正が行われる。この修正を行うのは、一般的にはフォトレタッチなどのアプリケーション１２ｄなどであるが、本画像処理システムではアプリケーション１２ｄが印刷処理を実行するときに出力するオブジェクトデータに対して、プリンタドライバ１２ｃが同修正処理を実行する。かかる印刷処理に際しては、アプリケーション１２ｄはオペレーティングシステム１２ａに対してオブジェクトデータを

出力することになる。

【００２１】図２は、アプリケーション１２ｄから出力されるオブジェクトデータである印刷イメージの処理単位の変化を示している。同図において、アプリケーション１２ｄにて生成された印刷イメージＰＩは、オペレーティングシステム１２ａに出力される。そして、オペレーティングシステム１２ａは描画命令群１２ａ１に基づいて、入力した印刷イメージＰＩを所定の描画命令に置き換えて印刷データを生成する。ここで、オペレーティングシステム１２ａが描画命令に置き換えた印刷データは、本来作画されたイメージをページ単位の座標をそのままにして形成されている印刷イメージＰＩを、処理の都合上、ランダムな矩形のオブジェクトデータに分割されている。そして、オペレーティングシステム１２ａは、この分割した矩形のオブジェクトデータをページ単位の座標とは無関係に、すなわち、印刷イメージＰＩの作画されたイメージに対応することなくランダムな位置に配置し、分割オブジェクトデータａ（１）～ａ（ｎ）を形成する。

【００２２】すなわち、それぞれの分割オブジェクトデータａ（１）～ａ（ｎ）において矩形で区分された領域の内部データは、座標が連続している領域に対応したデータを表わしているものの、相互に隣り合う分割オブジェクトデータａ（１）～ａ（ｎ）は座標に関しては何等連続性を有するものではない。従って、隣り合う分割オブジェクトデータａ（１）～ａ（ｎ）は不連続なデータとなっており、この形成された分割オブジェクトデータａ（１）～ａ（ｎ）の並び順に基づいて印刷を行っても、正常な印刷イメージＰＩを再現することは不可能になっている。このように分割オブジェクトデータａ（１）～ａ（ｎ）に分割された印刷イメージＰＩについて印刷を行うに場合、分割オブジェクトデータａ（１）～ａ（ｎ）は、プリンタドライバ１２ｃに入力されるとともに、適宜色調処理や拡大・縮小処理などの画像処理が実行される。そして、これらの画像処理の後に、カラープリンタ１７ｂが処理可能なラスタデータｂ（１）～ｂ（ｎ）を形成して、カラープリンタ１７ｄに出力する。

【００２３】かかる場合、プリンタドライバ１２ｃは、上述したラスタデータｂ（１）～ｂ（ｍ）をバンド単位で順次形成するために、ランダムに配置された分割オブジェクトデータａ（１）～ａ（ｎ）から相互に関連のある分割オブジェクトデータａ（１）～ａ（ｎ）を選択しスプールファイル１２ｃ１に格納する。そして、このスプールファイル１２ｃ１に格納した分割オブジェクトデータａ（１）～ａ（ｎ）から所定のデータ領域を抽出してメモリ１２ｃ２上に展開し、上述した色調処理や拡大・縮小処理などを実行する同プリンタドライバ１２ｃが備える画像処理を実行するモジュール１２ｃ３において利用可能にする。このメモリ１２ｃ２はハードウェア化されたプリンタドライバ１２ｃに配設されている所定の記憶領域であってもよいし、コンピュータ本体に取り付けられた図示しないＥＥＰＲＯＭなどの記録領域であってもよい。

【００２４】図２は、アプリケーション１２ｄから出力されるオブジェクトデータである印刷イメージの処理単位の変化を示している。同図において、アプリケーション１２ｄにて生成された印刷イメージＰＩは、オペレーティングシステム１２ａに出力される。そして、オペレーティングシステム１２ａは描画命令群１２ａ１に基づいて、入力した印刷イメージＰＩを所定の描画命令に置き換えて印刷データを生成する。ここで、オペレーティングシステム１２ａが描画命令に置き換えた印刷データは、オペレーティングシステム１２ａでの処理の都合上、複数の矩形からなるオブジェクトデータに分割される。そして、オペレーティングシステム１２ａは、この分割した矩形のオブジェクトデータをページ単位の座標とは無関係に、すなわち、印刷イメージＰＩの作画されたイメージに対応することなくランダムな位置に配置し、分割オブジェクトデータａ（１）～ａ（ｎ）を形成する。

【００２５】すなわち、それぞれの分割オブジェクトデータａ（１）～ａ（ｎ）において矩形で区分された領域の内部データは、作画座標が連続している領域に対応したデータを表わしているものの、相互に隣り合う分割オブジェクトデータａ（１）～ａ（ｎ）は座標に関しては連続性を有するものあるとは限られない。従って、隣り合う分割オブジェクトデータａ（１）～ａ（ｎ）は不連続なデータとなっており、この形成された分割オブジェクトデータａ（１）～ａ（ｎ）の並び順に基づいて印刷を行っても、正常な印刷イメージＰＩを再現することは不可能になっている。このように分割オブジェクトデータａ（１）～ａ（ｎ）に分割された印刷イメージＰＩについて印刷を行うに場合、所定の分割オブジェクトデータａ（１）～ａ（ｎ）がプリンタドライバ１２ｃに入力されるとともに、適宜色調処理や拡大・縮小処理などの画像処理が実行される。そして、これらの画像処理の後に、カラープリンタ１７ｂが処理可能なラスタデータｂ（１）～ｂ（ｎ）を形成して、カラープリンタ１７ｄに出力する。

【００２６】かかる場合、プリンタドライバ１２ｃは、上述したラスタデータｂ（１）～ｂ（ｍ）をバンド単位で順次形成するために、ランダムに配置された分割オブジェクトデータａ（１）～ａ（ｎ）から相互に関連のある分割オブジェクトデータａ（１）～ａ（ｎ）を選択してスプールファイル１２ｃ１に格納する。そして、このスプールファイル１２ｃ１に格納した分割オブジェクトデータａ（１）～ａ（ｎ）から上述した色調処理や拡大・縮小処理などの画像処理を実行する同プリンタドライバ１２ｃの備えるモジュール１２ｃ３が画像処理の処理

対象とするデータ領域を抽出し、同モジュール１２ｃ３が利用可能な状態でメモリ１２ｃ２上に展開する。

【００２７】次に、プリンタドライバ１２ｃがオペレーティングシステム１２ａに組み込まれた状態でのシステム上の概略構成を図３に示す。同図において、アプリケーション１２ｄから印刷要求があると、ＡＰＩ層から本プリンタドライバ１２ｃが提供する機能管理オブジェクトＰ１にアクセスされる。この機能管理オブジェクトＰ１は、分割画像管理オブジェクトＰ２によって上記分割オブジェクトデータａ（１）～ａ（ｎ）の管理を実行しつつ、処理画像管理オブジェクトＰ３によって処理画像を取得する。この処理画像を取得するにあたって機能管理オブジェクトＰ１は同処理画像管理オブジェクトＰ３に対して必要領域を要求するし、その処理の前提として予め分割オブジェクトデータａ（１）～ａ（ｎ）に関して分割画像登録の処理を実行する。

【００２８】すなわち、処理画像管理オブジェクトＰ３が実際の画像処理を実行する拡大処理オブジェクトＰ４と強調処理オブジェクトＰ５を介して分割画像管理オブジェクトＰ２に対して処理対象となる画像領域を通知するため、分割画像管理オブジェクトＰ２は既に登録されている分割オブジェクトデータａ（１）～ａ（ｎ）のうちの必要なものだけをメモリ１２ｃ２上に展開して利用可能とし、展開された結果を利用して強調処理オブジェクトＰ５と拡大処理オブジェクトＰ４が画像処理を実行することになる。従って、本発明にかかる画像処理プログラムにおいて、上述した機能管理オブジェクトＰ１と、分割画像管理オブジェクトＰ２と、処理画像管理オブジェクトＰ３とが本発明にかかるドライバを構成するとともに、拡大処理オブジェクトＰ４と、強調処理オブジェクトＰ５とが本発明にかかる画像処理を実行するモジュールを構成する。

【００２９】図４は入力される分割オブジェクトデータと取得される処理画像とを示している。同図において、元の画像をソース（Ｓｏｕｒｃｅ）画像と呼ぶとともに、拡大処理と強調処理を実施された画像をデスティネイション（Ｄｅｓｔｉｎａｔｉｏｎ）画像と呼ぶことにすると、入力される分割オブジェクトデータは９分割されたＳ－Ｕｎｉｔ．ａ～ｉである。ここで、デスティネイション画像は、一体として処理することも可能であるが、この場合も分割して処理するものとし、各分割画像をＤ－Ｕｎｉｔ．Ａ～Ｃと呼ぶ。Ｄ－Ｕｎｉｔ．Ａ～Ｃを得るためには処理前の画像領域を利用可能であることが必要であるが、入力されているのはＳ－Ｕｎｉｔ．ａ～ｉであって、全てを利用可能となっているか否かは分からない。従って、処理画像管理オブジェクトＰ３から分割画像管理オブジェクトＰ２に対して対応するＳ－Ｕｎｉｔ．ａ～ｉを利用可能とするように要求し、そのようにして利用可能にされるのがＳ－Ａｒｅａ．ａ～ｃである。ただし、このＳ－Ａｒｅａ．ａ～ｃは領域としては拡大前のＤ－Ｕｎｉｔ．Ａ～Ｃと一致するものではなく、周縁領域を含むものとなっている。このように周縁領域を含むのは、拡大処理オブジェクトＰ４と強調処理オブジェクトＰ５で画像処理する際に処理対象の画素だけでは処理が不可能であり、その周囲の画素も参照しなければならないからである。

【００３０】次に、各画像処理で参照する周縁領域を図５および図６に示す。具体的な処理内容については詳述しないが、図５に示す強調処理では対象画素を基準として上方と左方に１画素、右方と下方に２画素を参照する必要がある。また、図６に示す拡大処理では、対象画素を基準として右方左方と下方に１画素、上方に２画素を参照する必要がある。周縁領域を参照する必要がある具体的な演算例として、フィルタ処理が上げられる。この場合の強調処理と拡大処理の例にあてはめれば、強調処理においては４×４画素分のフィルタを利用し、拡大処理においては３×４画素分のフィルタを利用するので、周縁領域が必要になるといえる。図７は処理画像管理オブジェクトＰ３から分割画像管理オブジェクトＰ２に必要領域が通知される過程を示している。同図において、最終的に取得される処理画像の領域が、最初の必要領域Ｓ－Ａｒｅａ’として処理画像管理オブジェクトＰ３に要求されると、同処理画像管理オブジェクトＰ３は、拡大処理オブジェクトＰ４にこれを通知する。すると、拡大処理オブジェクトＰ４では自己の拡大倍率が２倍であるときには上記必要領域Ｓ－Ａｒｅａ’の元となる画像領域は１／２倍である。ただし、参照する画素は上方、右方、下方、左方に対してそれぞれ１、２、２、１画素づつであることを強調処理オブジェクトＰ５に通知する。この強調処理オブジェクトＰ５では拡大を行わないため、本来的に必要なのは拡大処理オブジェクトＰ４で必要とする画像領域であり、上記必要領域Ｓ－Ａｒｅａ’の１／２倍であるとともに、その周縁領域の（１，２，２，１）画素である。しかし、自己においても周縁領域を参照する必要があり、さらに周縁領域の（２，１，１，１）画素を加えて分割画像管理オブジェクトＰ２に通知する。すなわち、このような修正を加えた画像領域Ｓ－Ａｒｅａは必要領域Ｓ－Ａｒｅａ’の１／２倍の領域と、その周縁領域の（３，３，３，２）画素を加えた領域となる。むろん、このような周縁領域の大きさは拡大処理オブジェクトＰ４や強調処理オブジェクトＰ５の演算手法に依存しており、他の演算例では強調処理において上下左右の全てに２画素を要求しつつ、拡大処理で上方と左方に１画素、右方と下方に２画素を要求するというようなものもある。

【００３１】一方、分割画像管理オブジェクトＰ２は、機能管理オブジェクトＰ１を介して個々の分割オブジェクトデータが入力されており、領域としての情報を管理しながら実体についてはスプールファイルに格納して分割画像登録を実施している。スプールファイルはハード

ディスク１３ｂ上に格納されるため、他の処理オブジェクトは直接的に参照したり利用することはできない。しかしながら、この分割画像管理オブジェクトＰ２は以上のようにして通知された画像領域Ｓ－Ａｒｅａを他のオブジェクトから利用できるようにするため、メモリ上に展開する。この際、必要とされた画像領域Ｓ－Ａｒｅａのそのままをメモリ上に展開するのではなく、図８に示すように、あくまでも画像領域Ｓ－Ａｒｅａを包含するに足るＳ－Ｕｎｉｔ．ａ～ｉがどれにあたるかを調査し、該当する全てのＳ－Ｕｎｉｔ．ａ～ｉをメモリ上に展開する。ここで分割オブジェクトデータの数が少なければ画像領域Ｓ－Ａｒｅａを包含するＳ－Ｕｎｉｔ．ａ～ｉがどれであるかを判断するのは容易である。各Ｓ－Ｕｎｉｔ．ａ～ｉの実体データは、ハードディスク１３ｂ上に格納されるものの、各領域に関するヘッダ情報は、図９に示すようなテーブルに格納されている。各テーブルには次のＳ－Ｕｎｉｔ．ａ～ｉの情報開始位置を示すポインタｎｅｘｔ＿ｐｔｒが備えられている。本来であれば、このポインタをたどることによって全てのＳ－Ｕｎｉｔ．ａ～ｉのヘッダ情報を参照することができるが、各Ｓ－Ｕｎｉｔ．ａ～ｉが画像領域Ｓ－Ａｒｅａの一部とでも重なり合うか否かを判断するのはいくつかの比較演算を繰り返さなければならず、演算回数が多大となってしまう。一方、分割オブジェクトユニットの配置情報などから必ずしも全てのＳ－Ｕｎｉｔ．ａ～ｉと対比する必要もなく、現実的には参照すべきＳ－Ｕｎｉｔ．ａ～ｉは限られている。例えば、図９に示す例であれば、Ｓ－Ａｒｅａを展開するに際し、斜線を示すＳ－Ｕｎｉｔ．ａ～ｉだけを参照できればよく、一時的にではあるがこのテーブルにおけるポインタｎｅｘｔ＿ｐｔｒを振り替えておくようにしている。

【００３２】本実施形態においては、ポインタｎｅｘｔ＿ｐｔｒにより参照するＳ－Ｕｎｉｔ．ａ～ｉを振り替えて間接的にＳ－Ａｒｅａを形成するＳ－Ｕｎｉｔ．ａ～ｉを指定し、Ｓ－Ａｒｅａを高速に生成可能な構成を採用しているが、むろん、Ｓ－Ａｒｅａを高速に生成可能にする手法は限定されるものではない。図２１に示すようにＳ－Ａｒｅａの生成に必要なＳ－Ｕｎｉｔ．ａ～ｉをハードディスクと比較して相対的に高速アクセス可能な主メモリやキャッシュメモリなどのメモリに格納させ、Ｓ－Ａｒｅａの生成時に参照させるようにしてもよいし、図２２に示すように各Ｓ－Ｕｎｉｔ．ａ～ｉにＩＤ１～９を振り分けるとともに、このＩＤ１～９により順不同に配置されたＳ－Ｕｎｉｔ．ａ～ｉがソース画像の元画像を形成できるような順番に配置するＩＤテーブルを形成し、Ｓ－Ａｒｅａの生成時に参照するようにしてもよい。

【００３３】以上のようにして分割画像管理オブジェクトＰ２が画像領域Ｓ－Ａｒｅａを包含する全てのＳ－Ｕｎｉｔ．ａ～ｉをメモリ上に展開したら、強調処理オブジェクトＰ５が強調処理を実行し、次いで拡大処理オブジェクトＰ４が拡大処理を実行する。強調処理では対象画素の周縁領域として（２，１，１，１）画素が必要であり、強調処理の結果として生成される画像領域は図１０に示すように周縁領域として（２，１，１，１）画素を省いた破線内の領域となる。また、同様にして拡大処理では対象画素の周縁領域として（１，２，２，１）画素が必要であり、強調処理の結果として生成される画像領域は図１１に示すように周縁領域として（１，２，２，１）画素を省いた破線内の領域となる。むろん、拡大処理は内部に格子点を補間生成するものであるから、得られる処理画像は縦横２倍に拡大される結果、最初の必要領域Ｓ－Ａｒｅａ’と一致することになる。

【００３４】次に、分割画像管理オブジェクトＰ２がメモリ上にＳ－Ａｒｅａを展開する他の実施例を説明する。かかる実施例におけるプリンタドライバ１２ｃがオペレーティングシステム１２ａに組み込まれた状態でのシステム上の概略構成を図２３に示す。同図において、ＡＰＩ層から本プリンタドライバ１２ｃが提供する機能管理オブジェクトＰ１００にアクセスされ、この機能管理オブジェクトＰ１００は、分割画像管理オブジェクトＰ２００によって分割オブジェクトデータの管理を実行しつつ、処理画像管理オブジェクトＰ３００より拡大および色調に関する画像処理が実施された画像データを１ラインごと取得し、取得したライン単位の画像データを積み上げてデスティネーション画像を形成する画像データを生成する。

【００３５】具体的には、各モジュールにおいて色調処理オブジェクトＰ５００が分割画像管理オブジェクトＰ２００より引き渡された所定のライン単位の画像データに対して色調処理を実施し、所定のライン単位の画像データを形成し拡大処理オブジェクトＰ４００に引き渡す。次に、拡大処理オブジェクトＰ４００は、この色調処理が為された画像データに対して拡大処理を実施し、１ラインの画像データを形成し処理画像管理オブジェクトＰ４００に引き渡し、機能管理オブジェクトＰ１００は処理画像管理オブジェクトＰ３００より１ラインごとに画像データを取得しつつ積み上げてデスティネーション画像を生成する。

【００３６】ここで、本実施形態においては、拡大処理オブジェクトＰ４００は、１ラインの拡大処理後の画像データを生成するため、同１ラインと参照ラインとを含め５ラインの画像データを必要とする。従って、色調処理オブジェクトＰ５００に対して５ラインの画像データの引き渡しを通知することになる。また、色調処理オブジェクトＰ５００は、１ラインの色調処理後の画像データを生成するため、当該１ラインの画像データのみでよい。従って、分割画像管理オブジェクトＰ２００に対して１ラインの画像データの引き渡しを要求することになる。むろん、これらの処理にて必要とするライン数は、

特に限定されるものではなく、各オブジェクトの処理内容に応じて適宜変更可能である。

【００３７】次に、図２４に入力される分割オブジェクトデータと取得される処理画像とを示す。同図において、元の画像をソース（Ｓｏｕｒｃｅ）画像と呼ぶとともに、拡大処理と色調処理を実施された処理画像をデスティネイション（Ｄｅｓｔｉｎａｔｉｏｎ）画像と呼ぶことにする。また、入力される分割オブジェクトデータは９分割されたＳ－Ｕｎｉｔ．Ａ～Ｉである。ここで、デスティネイション画像は、一体として処理することも可能であるが、本実施例においてはライン単位に分割してラインごとに生成され、このラインごとに生成されたものを積み上げて最終的なデスティネーション画像が生成されるものとする。また、デスティネーション画像を分割するライン単位の各分割画像をＤ－ライン．０～ｎと呼ぶ。ここで、ｎは整数であり、生成されるデスティネーション画像の総ライン数を示している。最初に、Ｄ－ライン．０を得るために、処理画像管理オブジェクトＰ３００は、拡大処理オブジェクトＰ４００に対して１ラインの引き渡しを通知する。ここで、拡大処理オブジェクトＰ４００は、１ラインの拡大処理を実行するために、上述したように５ラインの画像データを必要とする。従って、次の色調処理オブジェクトＰ５００に対して５ラインの引き渡しを通知する。色調処理オブジェクトＰ５００は、色調処理を実行するにあたり１ラインの画像データにて処理可能であるため、分割画像管理オブジェクトＰ２００に１ラインの画像データの引き渡しを通知する。

【００３８】そして、分割画像管理オブジェクトＰ２００は、分割画像データＳ－Ｕｎｉｔ．Ａ～Ｉより所定の１ラインの画像データを形成し、色調処理オブジェクトＰ５００に引き渡す。色調処理オブジェクトＰ５００は、この引き渡された１ラインのデータに色調処理を実施し、色調処理が終わると、拡大処理オブジェクトＰ４００からは５ラインを要求されているため、５ライン分の色調処理が終わるまで、分割画像管理オブジェクトＰ２００に１ラインを引き渡しを通知し、分割画像管理オブジェクトは、要求に従い順次１ラインを生成し、色調処理オブジェクトに渡す。そして、色調処理オブジェクトＰ５００は、引き渡された１ラインの画像データに順次色調処理を実行する。色調処理オブジェクトＰ５００は、５ラインに対する色調処理が終わると、拡大処理オブジェクトＰ４００にこの５ラインを引き渡す。拡大処理オブジェクトＰ４００は、この引き渡された５ラインの内後ろ４ラインを参照して先頭１ラインの強調処理を実施する。そして、色調処理も強調処理も実施された１ラインのデスティネーション画像が生成され、処理画像管理オブジェクトＰ３００に引き渡される。

【００３９】最初の１ラインのデスティネーション画像が生成されると、以降、順次ｎラインまで、１ラインごとデスティネーション画像が生成される。この１ラインの画像を生成するにあたり、処理画像管理オブジェクトＰ３００は、拡大処理オブジェクトＰ４００に１ラインの引き渡しを通知する。ここで、拡大処理オブジェクトＰ４００は、既に、色調処理が終わっている５ラインを引き渡されている。従って、この５ラインのうち先頭の１ラインを不参照の扱いとし、２番目のラインについて拡大処理を実施するために、色調処理オブジェクトＰ５００に１ラインの引き渡しを通知する。この通知を受けると色調処理オブジェクトＰ５００は、分割画像管理オブジェクトＰ２００に１ラインの引き渡しを通知する。分割画像管理オブジェクトＰ２００は、Ｓ－Ｕｎｉｔ．Ａ～Ｉより所定の１ラインを形成し、色調処理オブジェクトＰ５００に引き渡す。色調処理オブジェクトＰ５００は、この引き渡された１ラインに対して色調処理を実施し、拡大処理オブジェクトＰ４００に引き渡す。拡大処理オブジェクトＰ４００は、この引き渡された１ラインを既に引き渡されている４ラインの最後尾に付加して、５ラインの処理領域を形成し、上述した２番目のラインについて拡大処理を実施する。この２番目の１ラインについて拡大処理が終わると、処理画像管理オブジェクトＰ３００に引き渡す。機能管理オブジェクトＰ１００は、このように処理画像管理オブジェクトＰ３００が取得した各オブジェクトにて各処理が実施されたライン単位の画像データを順次積み上げてデスティネーション画像を生成する。

【００４０】以上のようにして、最初に分割画像管理オブジェクトＰ２００が色調処理オブジェクトＰ５００に対して１ラインの画像領域をメモリ上に展開して引き渡し、色調処理オブジェクトＰ５００が色調処理を実行し、色調処理オブジェクトＰ５００において５ラインの画像データを生成すると、拡大処理オブジェクトＰ４００に引き渡し、拡大処理を実行する。そして、５ラインより生成された１ラインの画像データを処理画像管理オブジェクトＰ３００に引き渡す。そして、以降は、処理画像管理オブジェクトＰ３００，拡大処理オブジェクトＰ４００，色調処理オブジェクトＰ５００，分割画像管理オブジェクトＰ２００間にて１ラインの引き渡し通知および形成し、あるいは、画像処理を実行した１ラインの画像データを逆方向にて順次引き渡し、機能管理オブジェクトＰ１００にて積み上げることによりデスティネーション画像を生成する。本実施形態においては、分割画像管理オブジェクトＰ２００から色調処理オブジェクトＰ５００に引き渡される１ラインの画像データおよび色調処理オブジェクトＰ５００から拡大処理オブジェクトＰ４００に引き渡される５ラインあるいは１ラインの画像データがＳ－Ａｒｅａを構成することはいうまでもない。また、本実施形態においては、画像処理を実行する順番を色調処理オブジェクトＰ５００→拡大処理オブジェクトＰ４００としているが、むろん、これに限定さ

れるものではなく、拡大処理オブジェクトＰ４００→色調処理オブジェクトＰ５００としてもよいし、所定の設定により、拡大処理オブジェクトＰ４００あるいは色調処理オブジェクトＰ５００のいずれか一方のみを実行するようにしてもよい。

【００４１】ここで、図１２にプリンタドライバ１２ｃにおける手続の流れを示す。同図においては、オペレーティングシステム１２ａに組み込まれた状態でのプリンタドライバ１２ｃの概略構成を示しているが、図３ではプリンタドライバ１２ｃ単独での視点で示しており、さらにプリンタドライバ１２ｃ内を本来のドライバの機能に近い処理と、付加的な機能に近い処理とに分割し、左欄にドライバ機能をまとめ、右欄にモジュール機能をまとめている。すなわち、手順１においてはドライバ機能としてソース画像の取得処理があり、これは機能管理オブジェクトＰ１と分割画像管理オブジェクトＰ２の機能といえる。次に、手順２はフォトインスタンス取得処理があるが、これは各オブジェクト画像ごとに別個・並行に処理するために現在の実行処理対象を確認するためのものである。

【００４２】以上が前処理に対応し、処理の流れとしては手順３においてモジュールの側からドライバの側に向けて処理画像領域を上述したように通知する

一方、手順４では要求された処理画像領域を含むソース画像（Ｓ－Ｕｎｉｔ．ａ～ｉ）をドライバ側が登録し、これに続いて手順５では画像処理として強調処理および拡大処理を実行することになる。なお、画像処理は、オブジェクト画像全体についての評価を行っておき、その評価結果に基づいて処理を実行する。例えば、あるオブジェクト画像が全体的に暗いのであれば、画像処理で明るく修整すればよい。この場合、オブジェクト画像は分割オブジェクトデータに分割されてしまっているから、それぞれのオブジェクトデータをまとめて評価するか、個別的にオブジェクトデータについて評価しつつも、最終的には各オブジェクト画像ごとに評価をまとめる必要がある。

【００４３】本来、分割オブジェクトデータとオブジェクト画像との対応関係は付けられていないのが通常であり、この対応付けはプリンタドライバ１２ｃの側で行わなければならない。この対応付けの一つの手法として、仮想領域に分割オブジェクトデータを展開してオブジェクト画像を連結する手法があるし、別の手法として分割オブジェクトデータ同士の重なり合いや隣接具合から同じオブジェクト画像であるか否かを判断する手法がある。いずれの手法を採用するかはシステムの相違によってまちまちである。

【００４４】一方、上述したように画像処理のためにオブジェクト画像を評価するには、オブジェクト画像全体にわたって画素の情報をサンプリングすることが有効であり、簡易である。このようなサンプリングは、オブジェクト画像と分割オブジェクトデータとの対応関係が分かる前後のどちらで実行しても良い。すなわち、対応関係が分かる前に分割オブジェクトデータごとにサンプリングしておき、対応関係が分かってからサンプリング結果を統合することもできるし、対応関係が分かってから各オブジェクト画像ごとに対応する分割オブジェクトデータをサンプリングすることもできる。このようにサンプリングの実行タイミングは条件に応じてさまざまであり、図１２においても実行可能な三つのタイミングに対応して破線で示している。なお、ドライバの側で行うといってもサンプリング用のモジュールを用意することも可能である。

【００４５】手順５の画像処理は、このサンプリング結果を利用して実行され、生成された画像がデスティネーション画像となって手順６にて登録される。そして、登録されたデスティネーション画像は手順７でドライバ側にて出力されることになる。そして、最終的にデスティネーション画像が取得されたら手順８でソース画像をメモリから解放する。ここでソース画像の解放タイミングは特に制限されるものではない。例えば、一つのＳ－Ａｒｅａごとに登録・解放を実行するとすれば使用メモリ領域を低減できる反面、作業が煩わしい。これに対して使用メモリ領域の制限が少ない場合は、登録・解放の作業をこまめに行わなくても良いと言える。

【００４６】なお、上述した例では、デスティネーション画像を取得する意味で必要領域を通知しており、生成されるデスティネーション画像は任意のメモリ領域に展開されればよい。これに対してスプールファイル自体に対して拡大処理を実行したい場合もある。この場合は、拡大処理された画像をスプールファイルに対して付加し、元画像ファイルのポインタを拡大処理された画像ファイルにリンクさせればよい。例えばスプールファイルから読み出して仮想画像を構築し、画像処理を施してから元のスプールファイルを書き換えたいという処理があるとする。かかる場合に、拡大処理をスプールファイル中の画像ファイルに施してしまえば仮想画像上で拡大処理を実現しなくても良くなる。しかしながら、拡大処理すればスプールファイル中の画像ファイルよりも大きくなるから、拡大後の画像ファイルで書き換えることはできない。従って、拡大後の画像データをスプールファイルの後に付加していき、ポインタのリンクを張り替えるようにするのである。

【００４７】次に、拡大処理の具体的手法について説明する。一般に、ドットマトリクス状の画素からなる画像についてその格子間に新たに画素を生成する処理を拡大処理と呼んで広く実行されている。これは、ディスプレイ１７ａの解像度とカラープリンタ１７ｂの解像度に相違があるような場合、画素単位で割り当ててしまうと一定の大きさに表示されなくなるといった不具合を解消するためである。ここで、拡大処理には、次に示す手法が

知られている。
１．最近隣内挿法（単純水増しコピー）
２．３次たたみ込み補間（以下、３次補間という）
前者のものは数学的に最も高速な拡大手法であるし、後者のものはきれいな拡大結果を得られる反面で数学的には大量の演算を必要として時間がかかるという特徴を有している。いずれにしても、これらは画像処理の技術用語としてはフィルタ処理と呼ばれている。ここで、図１３や図１４は別々のフィルタ［１］，［２］を使用して拡大前の画像ａから拡大後の画像Ａを生成する原理を示している。この場合、拡大前の画像ａの数点を利用してフィルタ［１］，［２］にかけることにより、拡大後の画像Ａのある１点を求めることができる。従って、拡大後の画像Ａの全画素について本来的には演算を繰り返すことになる。
【００４８】ところで、フィルタ［１］，［２］には処理結果について差があるはずである。すなわち、前者は荒く見え、後者は滑らかに見えるように思われる。しかしながら、拡大結果を使用して印刷した場合、人間の目の認識の限界により必ずしもその差を認識することができ＊ない。すなわち、高度な演算を要するフィルタ［２］だけで拡大した場合と、フィルタ［１］，［２］を併用して拡大した場合とにおいて、拡大結果の差を認識できない場合がある。図１５は、後者の手順を示しており、元の画像ａに対してフィルタ［２］を利用して拡大処理を実行し、一度、画像ａ'を生成した後に、今度はフィルタ［１］を利用して拡大処理を実行し画像Ａを得ている。むろん、先のフィルタ［２］の拡大処理は３次補間を要するのでモジュールで実行し、フィルタ［１］の拡大処理はドライバで実現するようにしてもよい。この場合、もちろんフィルタ［１］とフィルタ［２］の拡大倍率によっては肉眼でも差が分かるが、そのような差が分からない範囲の組合せであるならば、処理時間のかかるフィルタ［２］にて生成する画像ａ'の面積は小さいため高速な処理を期待できる。なぜなら、拡大処理後の画像の面積に比例して演算量も多くなるからである。
【００４９】図１６は、３次補間の場合の既存の格子点と補間点との配置を示すとともに、数１は、この３次補間のフィルタ処理を行列演算式で示している。
【数１】

$$P = [f(Y_0)\ f(Y_1)\ f(Y_2)\ f(Y_3)] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} f(X_0) \\ f(X_1) \\ f(X_2) \\ f(X_3) \end{pmatrix}$$

なお、関数ｆ（ｔ）については、
ｆ（ｔ）　＝　｛ｓｉｎ（πｔ）｝／πｔ
で表され、ＵまたはＶをＷに置き換えることにより、
ｔ０　＝　１＋（Ｗ－｜Ｗ｜）
ｔ１　＝　　（Ｗ－｜Ｗ｜）
ｔ２　＝　１－（Ｗ－｜Ｗ｜）
ｔ３　＝　２－（Ｗ－｜Ｗ｜）
で表すことができる。
【００５０】このような３次補間演算は、一般的に多くの演算量を要することになるが、補間点の位置によっては高速化が可能であり、より具体的には整数倍の拡大に限れば処理量はかなり低減することができる。なお、ここでいう整数倍とは原点が格子線の交差点上に置かれる※
ｔ０　＝　１、　ｔ１　＝　０、　ｔ２　＝　１、　ｔ３　＝　２
となる。このとき、ｆ（ｔ）は、上式より以下のようになる。
ｆ(０)＝１、　ｆ(１)＝０、　ｆ(２)＝０
よって、Ｙ軸に関しては格子線上に存在する場合のＸ軸上の補間点は、
【数２】

$$P = [P_{01}\ P_{11}\ P_{21}\ P_{31}] \begin{pmatrix} f(X_0) \\ f(X_1) \\ f(X_2) \\ f(X_3) \end{pmatrix}$$

※ようになる倍率のことであって、拡大処理前後の格子点数の比を意味するものではない。図１７は、この定義でいうところの２倍の拡大であり、図１８は、３倍の拡大である。図中、大きな白丸が原点で、小さい黒丸が補間点である。これらの図から分かるように、整数倍の拡大においては、補間点のいくつかは格子線上に配置される。以下、具体的な高速化手段について説明する。
【００５１】（１）格子線上の補間点の演算簡易化
整数倍の拡大においては、補間点のいくつかは格子線上に配置される。従って、数１は、簡易化（演算量が減る）される。すなわち、補間点が格子線上に存在するということは、Ｗ（ＵあるいはＶのいずれか）が０になることを意味する。よって、
ｔ０　＝　１、　ｔ１　＝　０、　ｔ２　＝　１、　ｔ３　＝　２
によって算出することになり、Ｘ軸に関しては格子線上に存在する場合のＹ軸上の補間点は、
【数３】

$$P = [f(Y_0)\ f(Y_1)\ f(Y_2)\ f(Y_3)] \begin{pmatrix} P_{10} \\ P_{11} \\ P_{12} \\ P_{13} \end{pmatrix}$$

によって算出することになる。この数２、数３から明らかなように拡大処理の演算を簡易化することが可能になる。

【００５２】（２） 複数のフィルタを用意することによる高速化

整数倍のときに格子線上に位置する補間点はもっとも特異な例であるが、格子線上にない場合でも倍率を固定するとともに、補間点ごとのフィルタ処理を用意すれば、ｆ(t) は定数化できるので演算処理を高速化できる。かかる場合、ｆ（t） を定数化すると、

【数４】

$$P = [A\ B\ C\ D] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} K \\ L \\ M \\ N \end{pmatrix}$$

により表されることになる。ここにおいて、Ａ～Ｄ、Ｋ～Ｎ は、補間点の場所Ｗによって一意的に決定されうるものであり、予め演算してテーブルに保存しておけばよい。

【００５３】ここで、拡大処理において２倍の場合について図１７を参照してみると、局所的には４個の点を補間しなければならないので拡大処理前の画像ａを画像Ａに拡大する場合、４つのフィルタを用意することになる。これを図で表したものが図１９であり、複数のフィルタ処理を用意することを前提とすれば、ｆ（ｔ） の定数化が可能になる。ｆ（ｔ） の定数化が実現する具体的な高速化は以下のような理由による。まず、ｆ（ｔ） の定数化そのこと自体による高速化が可能である。なぜならば、ｆ（ｔ） の定数化は、ｆ（ｔ） で表せられる３次補間関数の演算の除去を意味するからである。次に、演算種別の変化によって高速化が可能である。一般に、ＣＰＵにとって同じ１回の掛け算でも、“変数 ＊ 変数” より “定数 ＊ 変数” の方が高速な処理となる。従って、数１と数４とを比較した場合、数４の方が高速化されることが分かる。さらに、一方の定数化によって “定数 ＊ 変数” のかけ算を演算テーブルにて代用することにより、高速化することができる。

【００５４】以上の高速化の効果をまとめると次のようになる。１つの補間点を求めるために、１つのフィルタ処理を行なう従来の手法によれば、

・３次補間関数の演算

・２０回の “変数 ＊ 変数” の演算

が必要である。しかしながら、上に述べたように複数のフィルタ処理を前提とした場合には、

・１６回の演算テーブルへのアクセス

・４回の “定数 ＊ 変数” の演算

により実現できる。従って、複数のフィルタを用意した方が高速化を実現できる。また、先に述べた格子線上の補間点の演算簡略化についても、同様にして複数のフィルタ処理を用意して実現可能である。そして、このような複数のフィルタ処理を用意することによる高速化と、格子線上にある補間点の式の簡略化による高速化は、整数倍であれば何倍でも有効である。

【００５５】（３）その他の高速化

（ａ）２のｎ乗倍の拡大における特異な補間点の演算式の簡略化

２のｎ乗倍の拡大においては（ｎは１以上の整数）、最近傍の原点と原点の２分の１にある場所の点を結んだ格子線（図１７の破線）に補間点が置かれるが、Ｘ軸上の場合は、

【数５】

X軸上

$$P = [A\ B\ C\ D] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} K \\ L \\ L \\ K \end{pmatrix}$$

により表され、Ｙ軸上の場合は、

※ ※【数６】

Y軸上

$$P = [A\ B\ B\ A] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} K \\ L \\ M \\ N \end{pmatrix}$$

により表されることになる。この場合、ベクトルの対称性を利用することにより、

・１６回の演算テーブルへのアクセス

・２回の “定数 ＊ 変数” の演算

だけで補間点を求めることができる。

【００５６】（ｂ） 各フィルタ間で共用できる演算の保存による演算量の節約

補間点数分のフィルタを用意することによって、高速化できることはすでに述べたが、各フィルタ間で共用できる演算を保存することによって演算量を節約することができる。

（ｃ）４倍拡大における高速化の例

図２０は、４倍拡大におけるコーディング例を示している。上述した高速化が具体的に適用されている行に注釈

を付している。すなわち、高速化は、
・原点のコピー
・格子線上の簡易式
・各フィルタ間で共用できる演算の保存
・２のｎ乗倍の特異点の簡易式
という手法で表れている。

【００５７】上述してきたように整数倍の補間拡大に限れば処理量はかなり低減することができる。従って、局所的に補間位置を特定することが高速化に有効であることが分かった。かかる整数倍による補間拡大の方法を適用する場合、整数倍の補間に限定せず、少数倍の補間拡大についても応用することが考えられる。ここでは、拡大処理の他の実施例について、少数倍、具体的には１．２５倍の補間拡大について考察する。この１．２５倍の補間拡大における局所的な作用を図２５に示す。同図において、局所的な１．２５倍とは、４点の原点を補間点の１点を含めて５点にすることである。すなわち、平面において、１６点の原点を補間点を含めて２５点に拡大することをいう。

【００５８】従って、図２５に示すように補間点の座標は予め推測することが可能であり、そのことによる演算の効率化および演算量の低減は上述した整数倍の補間拡大による実施例と同様である。よって、それぞれの補間点ごとに最適なフィルタを用意すればよいことになる。ここでは、それぞれの補間点ごとのフィルタを子フィルタと呼び、１６点の原点から補間点を含め２５点の画素を生成する。これらの子フィルタを使用した全体のフィ＊

＊ルタをフィルタ１６ｔｏ２５と呼ぶ。ここで、フィルタ１６ｔｏ２５を構成する子フィルタに必要なベクトル要素を示す。

・Ｐ１～Ｐ５を求めるために必要な１０２４倍されたベクトル要素
Ｐ１：｛０，１０２４，０，０｝
Ｐ２：｛ＶＡ，ＶＢ，ＶＣ，ＶＤ｝
Ｐ３：｛ＶＥ，ＶＦ，ＶＧ，ＶＨ｝
Ｐ４：｛ＶＨ，ＶＧ，ＶＦ，ＶＥ｝
Ｐ５：｛ＶＤ，ＶＣ，ＶＢ，ＶＡ｝

なお、ＶＡ～ＶＨは３次元補間関数とその補間位置から予め求められる定数である。

【００５９】このように、ある画像を１．２５倍にすることを考えた場合、フィルタ１６ｔｏ２５を１６点ずつ周期的に通すことになる。しかし、一般的にはフィルタ１６ｔｏ２５を通すことができない余りの領域が出ることは自明である。従って、フィルタ１６ｔｏ２５を適用する場合、参照画素の影響も考えなければならない。この余った領域は別のフィルタを通すことになるので、均一性がなくなり、歪みが生じる。ただし、これは人間の目の誤差の範囲内であることを前提とする。

【００６０】ここで、フィルタ１６ｔｏ２５を通した場合のソース画像とデスティネーション画像の大きさの関係を示す。ソース画像１辺の長さをｓＬとすると、フィルタ１６ｔｏ２５を通して得られるデスティネーション画像の１辺の長さｄＬは、以下のようになる。

・ｄＬ ＝ ｛（ｓＬ－１）／４｝＊５＋１　　｛ ｝は整数部分
・・・・式（１）

となる。ただし、複数の連続した局所的な箇所であっ※て、参照画素の影響がなければ、

・ｄＬ ＝ ｛ｓＬ／４｝＊５
・・・・式（２）

となる。

【００６１】次に、画像データを形成する画素数が４の倍数である場合に、１．２５倍の補間拡大を実施することについて検証する。具体的に、本実施形態においてはデジタルカメラの補間拡大などにより必要とされている、「１２８０画素＊９６０画素」を１．２５倍により★

・ｄＬ＿１＝[４＊Ｎ／４]＊５
＝[Ｎ]＊５
＝５＊Ｎ
・・・・（３）

となり、局所的な１．２５倍が利用可能であることが分かる。ただし、全体の大きさを考えた場合は、式（１）☆に基づいて考えなければならないので、

・ｄＬ＿２＝[４＊（Ｎ－１）／４]＊５＋１
＝[Ｎ－１／４]＊５＋１
＝５＊Ｎ－４
・・・・（４）
・・・・（４－１）
・・・・（４－２）

となる。式（４－２）は、デスティネーション画像として４画素足りない状態を示し、式（４－１）は（Ｎ－１）回しか、フィルタ１６ｔｏ２５を通していないことを示している。このことから、ソース画像でフィルタ１６ｔｏ２５を適用することができる画素は、４＊（Ｎ－１）点であり、４Ｎ－４（Ｎ－１）＝４点は未処理のままであることが分かる。また、デスティネーション画像としてフィルタ１６ｔｏ２５により取り出せる画素は、５＊（Ｎ－１）であり、５Ｎ－５（Ｎ－１）＝５点が生成できないことになる。従って、ソース画像の残りの４

★補間拡大し、「１６００画素＊１２００画素」に修正するということについて検証する。かかる場合、Ｘ座標およびＹ座標は、Ｎを整数とすると、４＊Ｎの画素数を５＊Ｎの画素数に変換することを示している。式（２）を使用して、ｓＬに４＊Ｎを代入すると、

点からデスティネーション画像における不足分５点を生成する方法を考える必要がある。

【００６２】このソース画像の残りの４点からデスティネーション画像における不足分５点を生成する方法を図２６示す。同図において、局所的には３点を４点にする、すなわち、４／３＝１．３３３３・・・・倍（以下、１．３倍）の補間拡大になる。ここで、この局所的な１．３倍の補間拡大に必要となるベクトル要素は以下のようになる。

・Ｐ１～Ｐ４を求めるために必要な１０２４倍されたベクトル要素

Ｐ１：｛０，１０２４，０，０｝

Ｐ２：｛ＶＫ，ＶＬ，ＶＭ，ＶＮ｝

Ｐ３：｛ＶＸ，ＶＹ，ＶＹ，ＶＸ｝

Ｐ４：｛ＶＮ，ＶＭ，ＶＬ，ＶＫ｝

なお、ＶＫ～ＶＮ，ＶＸ，ＶＹは３次元補間関数とその補間位置から予め求められる定数である。

【００６３】以上のことから、フィルタ１６ｔｏ２５で処理できなかった箇所については、Ｘ方向には、フィルタ１２ｔｏ２５Ｘ、Ｙ方向にはフィルタ１２ｔｏ２５Ｙを使用する。また、Ｘ，Ｙ方向ともに１．２５倍できない箇所はフィルタ９ｔｏ１６を使用する。ここで、このフィルタ１２ｔｏ１５Ｙを使用する際の原点と補間点の相関を示した相関図を図２７に示す（フィルタ１２ｔｏ２５Ｘは９０度回転した図になる。）。このように、１．２５倍の補間拡大では、局所的には１．２５倍と１．３倍の補間拡大を実行する。かかる場合、フィルタ１６ｔｏ２５、フィルタ１２ｔｏ２０Ｘ、フィルタ１２ｔｏ２０Ｘ、フィルタ９ｔｏ１６を使用すると補間拡大が可能となり、これに伴い補間拡大処理の高速化を実現することが可能になる。

【００６４】これらのフィルタで必要なベクトル要素をまとめると、１４個のベクトル要素が必要になる。よって、それぞれのベクトル要素に０～２５５を掛けた、２５６個の要素から形成される１４個の演算テーブルを使用することにより演算数を減らすことが可能になる。ここで、この演算テーブルの要素１個の大きさを４バイトとすると、１個の演算テーブルの大きさは、１０２４バイト（１ＫＢ）である。これが１４個あるので、演算テーブルに必要なメモリは１４ＫＢとなる。

【００６５】このように、ソース画像の取得処理後（手順１）、フォトインスタンス取得処理（手順２）を経たら、モジュールの側からドライバの側に向けて処理画像領域を通知するが（手順３）、このときに各画像処理モジュールで必要な周縁領域を加えていくことにより、要求された処理画像領域に必要な周縁領域を加えた上でソース画像（Ｓ－Ｕｎｉｔ）をドライバの側が登録し（手順４）、これを利用して順次画像処理（強調処理と拡大処理（手順５））を実行することになり、画像処理が周縁領域を必要とする場合にも極めてシンプルに必要領域を確保して処理を実行することができる。

【図面の簡単な説明】

【図１】本発明の一実施形態にかかる画像処理プログラムを実行するコンピュータシステムのブロック図である。

【図２】印刷イメージの処理単位の変化を示す図である。

【図３】オペレーティングシステムに組み込まれた状態でのプリンタドライバの概略構成を示す図である。

【図４】画像の処理単位を示す図である。

【図５】強調処理で必要な周縁領域を示す図である。

【図６】拡大処理で必要な周縁領域を示す図である。

【図７】モジュール側からドライバ側に必要領域を通知する過程を示す図である。

【図８】Ｓ－ＡｒｅａとＳ－Ｕｎｉｔの関係を示す図である。

【図９】Ｓ－Ｕｎｉｔのポインタテーブルを示す図である。

【図１０】強調処理を実行する場合の実質的処理対象領域を示す図である。

【図１１】拡大処理を実行する場合の実質的処理対象領域を示す図である。

【図１２】プリンタドライバにおける手続の流れを示す図である。

【図１３】第１のフィルタ処理による拡大処理を示す図である。

【図１４】第２のフィルタ処理による拡大処理を示す図である。

【図１５】二段階のフィルタ処理による拡大処理を示す図である。

【図１６】３次補間の格子点と補間点の関係を示す図である。

【図１７】２倍補間の格子点と補間点の関係を示す図である。

【図１８】３倍補間の格子点と補間点の関係を示す図である。

【図１９】４つのフィルタ処理を利用した３次補間による拡大処理を示す図である。

【図２０】４倍拡大におけるコーディング例を示す図である。

【図２１】Ｓ－Ｕｎｉｔについて処理対象となるものを分離格納した場合の図である。

【図２２】Ｓ－Ｕｎｉｔの処理領域テーブルを示す図である。

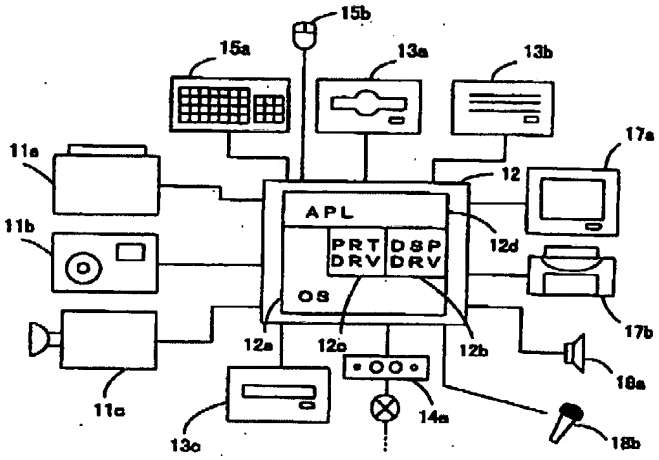
【図２３】他の実施の形態におけるオペレーティングシステムに組み込まれた状態でのプリンタドライバの構成を示す図である。

【図２４】他の実施の形態おける画像の処理単位を示す図である。

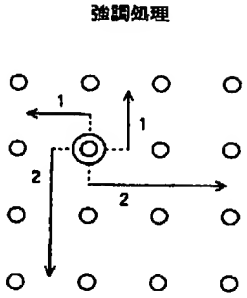
【図２５】１．２５倍補間の格子点と補間点の関係を示

す図である。
【図２６】１．３倍補間の格子点と補間点の関係を示す図である。
【図２７】フィルタ１２ｔｏ１５Ｙを使用する際の原点と補間点の相関を示した相関図である。
【符号の説明】
１０…コンピュータシステム
１１ａ…スキャナ
１１ａ２…スキャナ
１１ｂ…デジタルスチルカメラ
１１ｂ１…デジタルスチルカメラ
１１ｂ２…デジタルスチルカメラ
１１ｃ…ビデオカメラ
１２…コンピュータ本体
１２ａ…オペレーティングシステム
＊１２ｂ…ディスプレイドライバ
１２ｃ…プリンタドライバ
１２ｄ…アプリケーション
１３ａ…フロッピーディスクドライブ
１３ｂ…ハードディスク
１３ｃ…ＣＤ－ＲＯＭドライブ
１４ａ…モデム
１４ａ２…モデム
１５ａ…キーボード
１５ｂ…マウス
１７ａ…ディスプレイ
１７ｂ…カラープリンタ
１８ａ…スピーカ
１８ｂ…マイク
＊

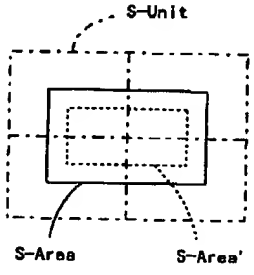
【図１】



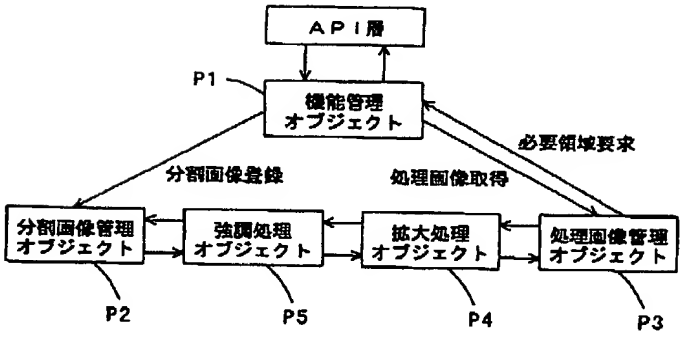
【図５】



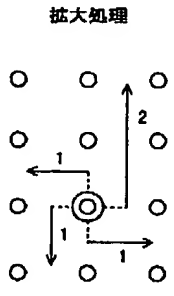
【図８】



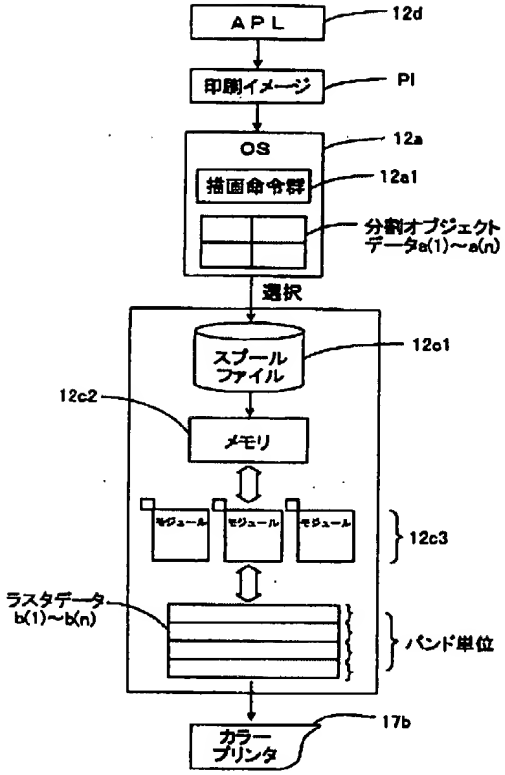
【図３】



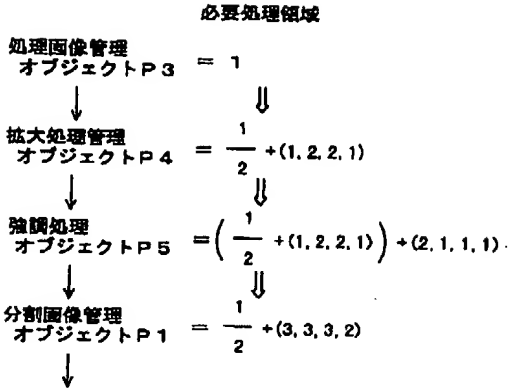
【図６】



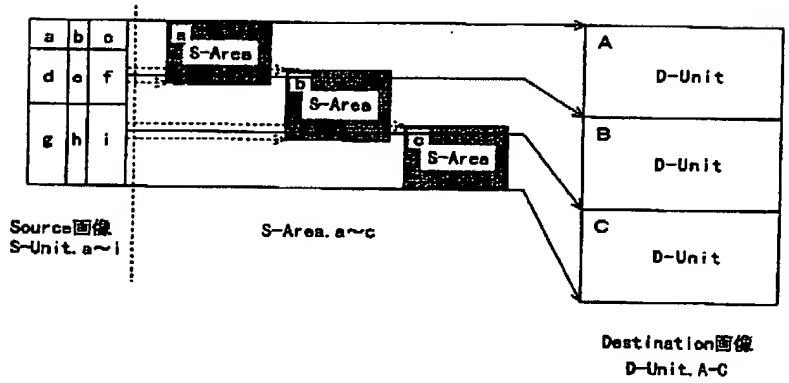
【図２】



【図７】

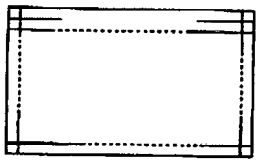


【図４】



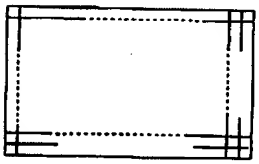
【図１０】

強調処理実行

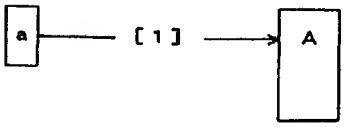


【図１１】

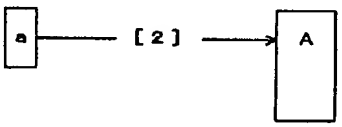
拡大処理実行



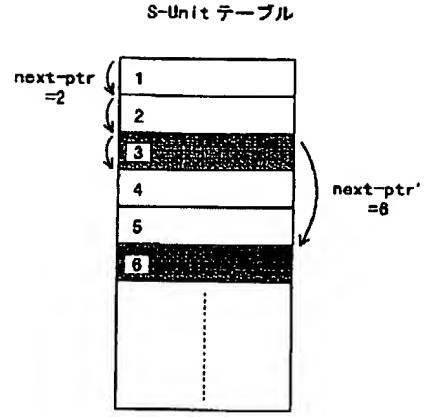
【図１３】



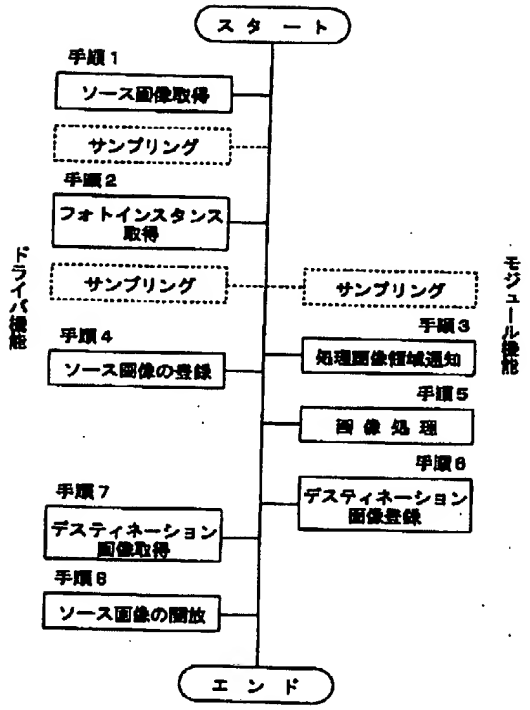
【図１４】



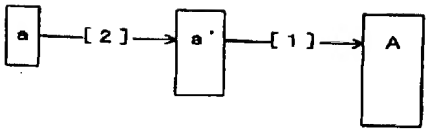
【図９】



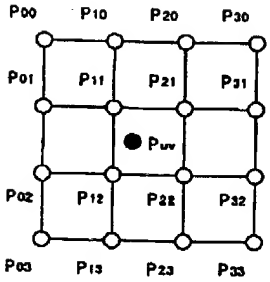
【図１２】



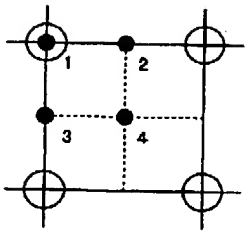
【図１５】



【図１６】

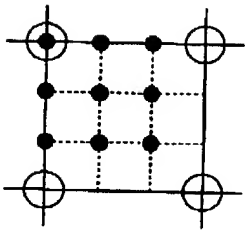


【図１７】



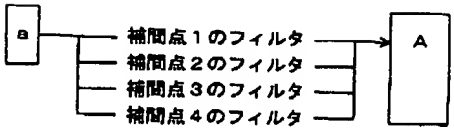
２倍補間

【図１８】



３倍補間

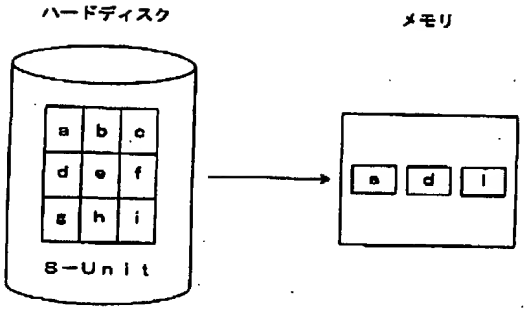
【図１９】



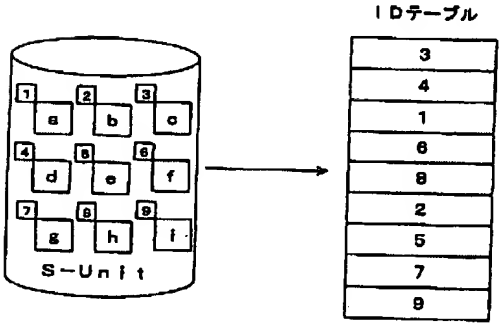
【図２０】

```
// ey= 0, ex=0 ********************原点のコピー
dP = P11;
// ey=0, ex=1 ************************格子線上の簡易式
dP = (T10[P10]+T11[P11]+T12[P12]+T13[P13]+512)>>10;
// ey=0, ex=2 ************************格子線上の簡易式
dP = (T20[P10]+T21[P11]+T22[P12]+T23[P13]+512)>>10;
// ey=0, ex=3 ************************格子線上の簡易式
dP = (T30[P10]+T31[P11]+T32[P12]+T33[P13]+512)>>10;

for( ey = 1; ey < 4; ey++ ) {
   各フィルタ間で共用できる演算の保存
   switch( ey ) {
   case 1:
      for( i = 0; i < 4; i++ ) {
         Px[i] = (T10[P0i]+T11[P1i]+T12[P2i]+T13[P3i]);
      }  break;
   case 2:
      for( i = 0; i < 4; i++ ) {
         Px[i] = (T20[P0i]+T21[P1i]+T22[P2i]+T23[P3i]);
      }  break;
   case 3:
      for( i = 0; i < 4; i++ ) {
         Px[i] = (T30[P0i]+T31[P1i]+T32[P2i]+T33[P3i]);
      }  break;
   }
   // ey=n, ex=0 ********************格子線上の簡易式
   dP = (Px[1]+512) >> 10 ;
   // ey=n, ]ex=1 ********************
   dP =(Px[0]*M40+Px[1]*M41+Px[2]*M42+
                                   Px[3]*M43+524288) >> 20;
   // ey=n, ex=2*****************2のn乗倍の特異点の簡易式
   dP =(Px[0]+Px[3]*M44+PX[1]*+[2])*M45
                                   +524288) >> 20;
   // ey=n, ex=3 ********************
   dP =(Px[0]*M43+Px[1]*M42+Px[2]*M41+
                                   Px[3]*M40+524288) >> 20;

}
}
```

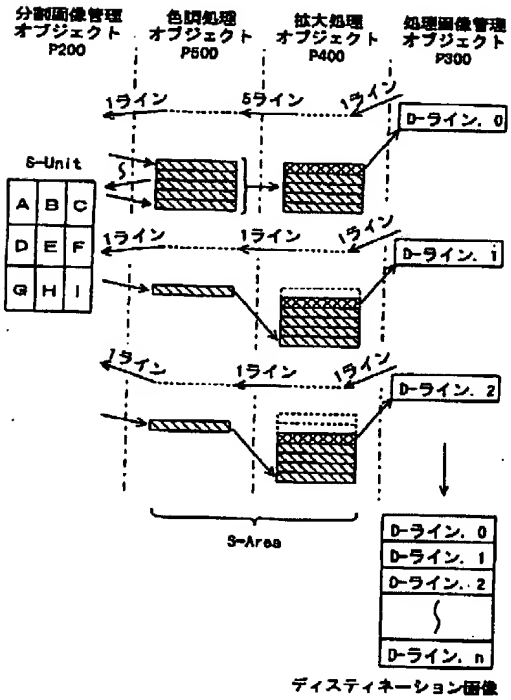
【図２１】



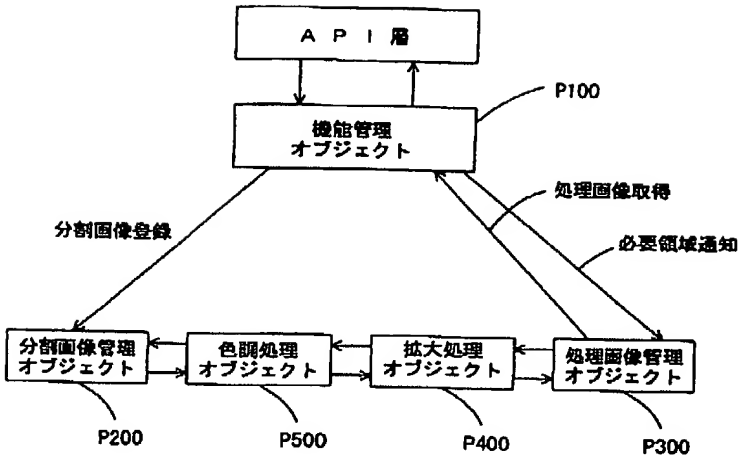
【図２２】



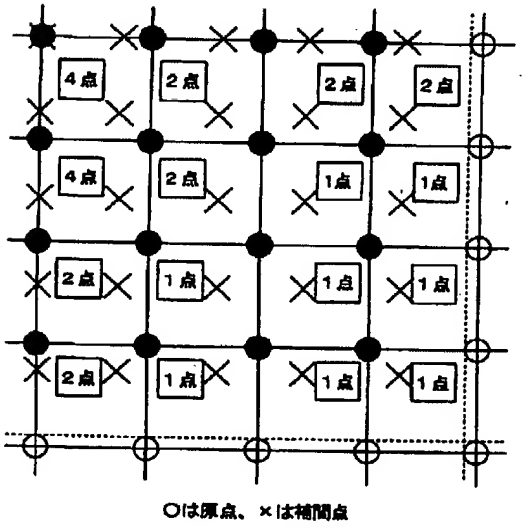
【図２４】



【図２３】


ＡＰＩ層
P100
機能管理
オブジェクト
処理画像取得
分割画像登録
必要領域通知
分割画像管理
オブジェクト
色調処理
オブジェクト
拡大処理
オブジェクト
処理画像管理
オブジェクト
P200
P500
P400
P300

【図２５】


4点
2点
2点
2点
4点
2点
1点
1点
2点
1点
1点
1点
2点
1点
1点
1点
○は原点、×は補間点

【図２６】

【図２７】